Quantified Boolean Formulas Satisfiability Suggested Format - DRAFT -

Last Revision July 3, 2001

This paper outlines a suggested format for quantified Boolean formula satisfiability problems. If you have any comments on this format or you have information that you think should be included, please send a note to qbf@dist.unige.it

1 Introduction

In the last few years we have witnessed the birth of a large number of new solvers for quantified Boolean formulas (QBF). Currently, every solver has a different input syntax. This paper aims to suggest a standard format to present problems to a QBF solver. Since QBF syntax extends the Satisfiability (SAT) syntax, a natural choice for the QBF format is an extension of the DIMACS standard for SAT problems. In this way a QBF solver can take as input a SAT problem in DIMACS format and solve it. In section 2 we show the syntax of QBF in Clausal Normal Form (CNF) and in section 3 we outline the extension of the DIMACS format for SAT to handle QBF: we call Q-DIMACS this extended format.

2 QBF in CNF-format

Consider a set P of propositional letters. An *atom* is an element of P. A *literal* is an atom or the negation of an atom. A *clause* C is a p-ary ($p \geq 0$) disjunction of literals. A *propositional formula* is an m-ary ($m \geq 0$) conjunction of clauses. As customary, we represent a clause as a set of literals, and a propositional formula as a set of clauses.

A *QBF* is an expression of the form

$$Q_1 x_1 \ldots Q_n x_n \Phi, \qquad (n \geq 0) \tag{1}$$

where every Q_i ($1 \leq i \leq n$) is a quantifier, either existential $\exists$ or universal $\forall$; $x_1, \ldots, x_n$ are pairwise distinct atoms in P; and Φ is a propositional formula in

the atoms $x_1, \dots, x_n$. $Q_1 x_1 \dots Q_n x_n$ is the *prefix* and Φ is the (quantifier-free) *matrix* of (1).
So an examples of QBF should be :

Example 1:

$\forall\ x_1\ \exists\ x_2\ \forall\ x_3\ \exists\ x_4\ x_5$
$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_1 \vee x_3 \vee x_4) \wedge$
$(x_1 \vee \neg x_4 \vee x_5) \wedge (\neg x_1 \vee x_2 \vee x_5) \wedge$
$(x_1 \vee \neg x_3 \vee x_4 \vee \neg x_5) \wedge (\neg x_1 \vee x_3\ \vee \neg x_4) \wedge$
$(\neg x_1 \vee \neg x_2 \vee \neg x_3 \vee \neg x_5) \wedge (x_1\ \vee \neg x_4) \wedge (x_3 \vee \neg x_2 \vee x_1)$

3 Q-DIMACS Syntax

The *QBF problem* is :
Given a QBF formula is it True?
To represent an instance of such problem we will create an input file that contains all the informations needed to define a QBF problem. The input file will be an ASCII file consisting of the following three sections.

Preamble : The Preamble contains information about the instance. This information is contained in lines. Each line begins with a single character (followed by a space) that determines the type of the line. These types are as follows:

- **Comments**: comment lines give human-readable information about the file and are ignored by programs. Comment lines appear at the beginning of the preamble. Each comment line begins with a lowercase character `c`. Example :

  ```
  c This is an example of a comment line
  ```

- **Problem line**: there is only one problem line per input file. The problem line must appear before the prefix and after any comment line. The problem lines has the following format:

  ```
  p cnf VARIABLES CLAUSES
  ```

  The lower case string `p cnf` means that this is the problem line and the instance is a cnf formula. The `VARIABLES` field contains an positive integer value specifying the total number of variables in the instance. The `CLAUSES` field contains an integer value specifying the number of clauses in the instance. This line must be occur as the last line of the preamble.

Prefix : The prefix encode the information about the quantifiers and the way they are applied to proposition. Every line begins with a letter followed by a set of positive integers.
The letters are

- *a* if the line represents an universal quantifier list,
- *e* if the line represents an existential quantifier list.
- *r* if the line represents a random quantifier list.

Every line ends with 0.
For example refer to Example (1) the prefix should be:

```
a 1 0
e 2 0
a 3 0
e 4 5 0
```

Matrix : The matrix represents the clauses and it appears immediately after the prefix lines. The variables are assumed to be numbered from 1 up to `VARIABLES`. It is not necessary that every variable appear in an instance. Each clause will be represented by a sequence of integers, which are separated by either a space, a tab, or a new line character. The non negated version of a variables is represented by `i`; the negated version is represented by `-i`. Each clauses is terminated by a value 0. This format allows clauses to be on multiple lines.

Example : Using the example (1) a possible input file would be

```
c Example CNF format file
c
p cnf 5 9
a 1 0
e 2 0
a 3 0
e 4 5 0
1 3 4 0
-1 3 4 0
1 -4 -5 0
-1 2 5 0
1 -3 4 -5 0
-1 3 -4 0
-1 -2 -3 -5 0
1 -4 0
3 -2 1 0
```

Every variable that does not show up in the prefix and occurs in the matrix is intended to be existentially quantified in the outermost position of the prefix.

3.1 Compatibility with DIMACS syntax

Given our definition of Q-DIMACS format a Q-DIMACS file without the prefix part is a DIMACS file and it is interpreted as a Q-DIMACS file where all the variables are quantified existentially. In this way a QBF solver can read a satisfiability instance and solve it.